TURBOIMAGE RUN-TIME OPTIONS: BALANCING PERFORMANCE WITH DATA BASE INTEGRITY

Author: Peter Kane, Information Technology Group,
Hewlett-Packard, Cupertino, California, U.S.A.

Summary

Along with improvements in the areas of performance and data base limitations, run-time options were added to TurboIMAGE to allow more flexibility between having high performance and high recoverability. A run-time option is an option which can be used without changes made to any application. In the case of TurboIMAGE, all of the options I will discuss can be enabled through DBUTIL. The purpose of this paper is to compare all combinations of these options, considering the performance impact and what recovery is available.

Introduction

In IMAGE/3000, two run-time options, LOGGING and Intrinsic Level Recovery (ILR) are available. A user can enable LOGGING, ILR, both, or neither. There are two differences between these options. The first is that ILR guarantees only physical integrity, i.e. no broken chains, while using logging and DBRECOV guarantees both physical and logical integrity, i.e. only finished transactions appear in the data base after recovery. The second difference is that with ILR, recovery happens automatically on the first DBOPEN of the data base following an interrupted DBPUT or DBDELETE, while using DBRECOV means restoring an old copy of the data base and waiting while all finished transactions in the log file are issued. Whatever the wait on DBRECOV, it is a much less lengthy process than recovery of a data base not using either ILR or logging, which is by DBUNLOAD and DBLOAD (or a third party utility).

Another option which is not a run-time option since it means application changes, is output deferred mode. Output deferred can be enabled in IMAGE/3000 only when the data base is opened in mode 3 (exclusive modify access), and is enabled by calling DBCONTROL with mode 1. Output deferred can therefore be used only in single-user environments. This mode can drop significantly the elapsed and CPU times needed by DBPUT, DBDELETE, and DBUPDATE. The reason is that modified buffers and set labels are not written to the data base until either the buffer is needed to hold a different data block, or else a DBCLOSE mode 1 or 2 is issued. The drawback of output deferred can be inferred by this last point, which is that a system failure occurring in the middle of an output deferred application can leave a badly damaged data base. The usual use for output deferred is in a batch environment, where the data base is stored before running the applications. If the system fails while the applications are running, the stored data base would be restored and the applications would then be restarted. Output deferred mode can be used in an interactive environment, if logging is set up (otherwise a DBUNLOAD and DBLOAD is necessary if the system fails). However, exclusive access is required, which usually eliminates output deferred for consideration in an interactive environment.

The above options allow some flexibility, but some issues have been outstanding:

1. If there are a lot of transactions on a log file, recovery to achieve logical integrity can mean a lot of down-time.

2. In shortening the maintenance cycle so that recovery can take less time, the data base must be stored more often. Users can not issue transactions against the data base during these times.

3. Applications using output deferred mode must be operating exclusively. Therefore batch processing can take longer than necessary.

4. A high volume of modifications can have a major impact on performance.

Therefore two more run-time options have been added to TurboIMAGE as answers to the above issues. These are: AUTODEFER, or multi-user output deferred mode, and ROLLBACK, or Rollback recovery which allows a faster recovery than IMAGE's recovery method (which I refer to as Roll Forward recovery). The options which have been available in IMAGE have all been carried over to TurboIMAGE, giving four different run-time options. These options can be used in seven different combinations to balance performance and integrity. In this paper I will discuss each combination, specifying the advantages and disadvantages and my recommendations for its use.

Combinations Available with TurboIMAGE

The following is a list of the possible combinations:

1. No options used
2. LOGGING enabled
3. AUTODEFER enabled
4. AUTODEFER and LOGGING enabled
5. ILR enabled
6. ILR and LOGGING enabled
7. ROLLBACK enabled

It may noted that some options are not compatible with others, for instance AUTODEFER and ILR are not compatible. It also should be noted that enabling ROLLBACK automatically enables LOGGING and ILR.

Each of these combinations will now be looked at separately.

Combination 1: No options used.

Performance: Modified buffers and labels are written to disc (or cache if caching is enabled and BLOCKONWRITE is set to NO), within the intrinsic which did the modification. This means that modify intensive applications must wait frequently for writes to occur. It also means that buffers are never left dirty after an intrinsic finishes, which has a positive impact on read intensive environments (more on this in the discussion of AUTODEFER).

Integrity: If the system fails during processing, physical corruption may result. If this happens, a DBUNLOAD and DBLOAD (other utilities from third parties may be used instead) is necessary to recover the data base. Note that logical recovery is not possible in this case. If disc caching is used and BLOCKONWRITE is set to NO, multiple corruptions can result from a single system failure.

Advantages: The multi-user, read intensive environment probably sees the best performance with this combination. The overhead of logging and ILR is not seen.

Disadvantages: Very lengthy recovery if a system failure has caused physical corruption.

Combination 1 (continued)

Recommendation: Use for read intensive (approximately 80% reads, 20% modifications) application mixes on non-critical data bases. Modify intensive environments can be improved in performance by using other options.

Combination 2: LOGGING enabled.

Performance: Contrary to what many users believe, logging causes only a slight (ranging from about 3% to 8%) degradation in performance. The higher end of this range is usually seen in the modify intensive environments. The reason the degradation is slight is because with only logging enabled log writes stay in memory until a DBEND or a DBCLOSE is issued, or if the log buffer in memory fills up. Therefore this combination is only slightly lower in performance than using no options at all.

Integrity: Physical integrity can be achieved without using the lengthy process of DBUNLOAD and DBLOAD. Logical integrity is possible if the applications are written using DBBEGINs and DBENDs. By logging to tape, disc failures can be recovered from.

Advantages: Physical recovery is far less lengthy than DBUNLOAD and DBLOAD. User specific data can be obtained from the log file. Logging to tape or to a different disc from the data base can provide recovery from media failures.

Disadvantages: Dedicated tape drive or usage of disc space for log records. To attain logical recovery, applications must have DBBEGIN and DBEND calls to define logical transactions. Periodic down-time is necessary to back up the data base and start a new log cycle.

Recommendation: Probably best use is in read intensive environments where logical transactions have been defined, where logical recovery is desired, and where application performance is more of an issue than down-time required to recover from a failure. Also provides best protection against media failures.

Combination 3: AUTODEFER enabled.

Performance: Usually will prove to allow the best performance, especially in modify intensive environments. Dirty buffers and labels are not flushed to disc (or cache) until DBCLOSE mode 1 or 2, or unless a buffer is needed to hold a different data block from the data base and all other buffers are dirty. From this one can see that TurboIMAGE will try to keep dirty buffers in memory as long as possible, in an effor to eliminate unnecessary writes to disc. This is useful if users are modifying the same buffers over and over again. However, if one user modifies a buffer containing a data block no other user ever accesses, that block may stay in its buffer for a long time. This will mean less buffers for the other users to do reads, which will in turn mean that buffers may be overlayed with other data blocks before the original user is through. This has an impact on the read intensive environment where there are occasional modifications. In the modify intensive

Combination 3 (continued)

Performance: (continued)	environments this is not much of an issue because all of the buffers are modified in time.
Integrity:	In short, NEVER use AUTODEFER by itself in interactive environments. This is because the user never has any idea whether the modified blocks or labels (which contain data set ends of file, delete chain heads, etc.) have made it to disc until DBCLOSE time. A DBUNLOAD/DBLOAD is necessary to recover anything at all if the system fails while applications are running. AUTODEFER is fine with batch processing, if a store of the data base is done first. Then if the system fails, the data base could be restored and the applications redone.
Advantages:	Highest performance in applications which do more than an occasional modification.
Disadvantages:	Physical integrity is highly at stake. Logical recovery not possible at all.
Recommendation:	Batch processing where applications modify the data base more than occasionally.

Combination 4: AUTODEFER and LOGGING enabled.

Performance:	Since the log writes are buffered by MPE until the buffer fills or until DBCLOSE or DBEND, logging adds very little performance overhead in this combination as opposed to having AUTODEFER alone. The performance advantages of AUTODEFER are still realized with this combination.
Integrity:	Roll forward recovery is available. Therefore, physical integrity can be achieved, while logical integrity can be achieved if transactions have been defined in the applications using DBBEGINs and DBENDs.
Advantages:	High performance in applications which do more than occasional modifications along with a recovery method in case of a system failure. May be the best combination for environments which are CPU bound.
Disadvantages:	Roll forward recovery is not the fastest recovery method. Down-time is necessary to back up the data base and start a new log cycle. Dedicated tape drive or disc space is necessary.
Recommendation:	Use in interactive environments where application performance is highest concern, and where data base modifications are done more than occasionally.

Combination 5: ILR enabled.

Performance:	Performance degradation with modifications, for two reasons. The first is because there are at least two additional writes to disc to update the ILR file for each DBPUT and DBDELETE. The second reason is that all writes to the data base and to the ILR file go through the Serial Write Queue. TurboIMAGE calls FSETMODE to set this file system option if it determines that ILR is enabled. Going through the Serial Write Queue means that writes can not operate concurrently if they are to different discs.
Integrity:	Automatic physical recovery on the first DBOPEN of the data base following an interrupted DBPUT or DBDELETE. ILR has been enhanced to redo the interrupted intrinsic rather than rolling it out as in IMAGE. No logical recovery is available. ILR alone does not protect against media failures.
Advantages:	Quick physical recovery method, which is automatic. Easy to use.
Disadvantages:	Performance degradation for applications using a high number of DBPUTs and DBDELETEs. No logical recovery available. Can not recover from media failures. Not compatible with AUTODEFER.
Recommendation:	For environments with a low volume of DBPUTs and DBDELETEs, this is an inexpensive and painless way of insuring physical integrity. For applications without DBBEGINs and DBENDs, this may be more useful than enabling ROLLBACK.

Combination 6: ILR and LOGGING enabled.

Performance:	Logging will cause a slight amount of degradation over having ILR alone enabled.
Integrity:	Can achieve quick and automatic physical recovery with ILR, and can have logical recovery with DBRECOV. Media failures can be recovered from.
Advantages:	Quick physical recovery method. Logical recovery available. Recovery from media failures is available.
Disadvantages:	Performance degradation due to ILR during DBPUTs and DBDELETEs. Logging maintenance.
Recommendation:	Probably best use is in environments where ILR's automatic recovery is the usual recovery method, and where logging is used to protect from media failures. For environments where logical recovery is possible (DBBEGINs and DBENDs are used), ROLLBACK is probably a better option.

Combination 7: ROLLBACK enabled (ILR and LOGGING will also be enabled).

Performance:	This combination has some more degradation over Combination 6. This is because log writes will now be written directly to disc using BLOCKONWRITE instead of being buffered by MPE as in all of the other combinations using logging. Furthermore, since all data base and ILR writes on this option will go through the Serial Write Queue, the log write will have to wait for any or all of the previous writes before it can be written itself. Therefore, DBPUT and DBDELETE should see the most performance degradation from this method.
Integrity:	Rollback recovery used for system failure or "soft crash". Roll forward recovery can be used if there has been a media failure or "hard crash". Logical and physical recovery are both available.
Advantages:	Quick logical and physical recovery (rollback is much faster than roll forward recovery). Stores of the data base do not have to be taken as often, since the length of the log file is not as great of an issue with rollback.
Disadvantages:	Performance degradation, especially on DBPUTs and DBDELETEs. DBUPDATEs also affected.
Recommendation:	Use where transactions have been defined and where it is crucial that the data base be available, and logically intact.

Biography

Peter Kane
has been with Hewlett-Packard for the last 3 1/2 years. He is an Online Support Engineer for Data Base products, which includes IMAGE/3000 and TurboIMAGE/3000. He in the past was responsible for SE training on the same products.